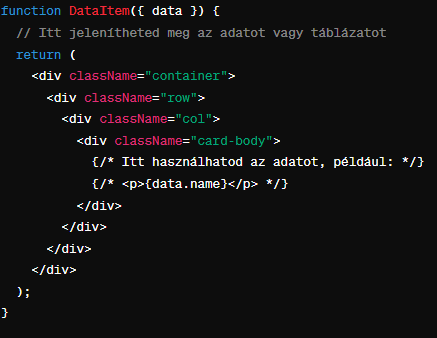
**Frontend útmutató**

***JS React***

1. **Projekt létrehozása**
   * **npm create vite@latest**
   * Add name
   * Select framework
   * Select langauge
   * projekt mappájában:
     + **npm install**
     + **npm install bootstrap**
     + **npm run dev**
   * Kezdeti környzet kialakítása
     + App.css tartalának törlése
     + index.css tartalmának törlése
     + App.jsx tartalmának törlése
2. **Kinézet kialakítása:**
   * Felesleges fájlok eltávolítása az **App.jsx** fájlból, hogy egy tiszta komponenst kapjunk, amely csak egy üres JSX visszatérítést tartalmaz.
     + **function App(){return (<></>)} export default App;**
   * Bootstrap beimportálása:
     + **import** ’bootstrap/dist/css/bootstrap.min.css’;
     + class helyett : ClassName
   * HTML kód megírása
   * Index.html fájlban a cím átadása <title> tag-be
   * Az adatokat legalább egy külön komponensbe kiszervezni
     + pl.: **DataItem.jsx**



* + Prop validálás:
    - ***A képen szöveg, képernyőkép, Betűtípus látható

      Automatikusan generált leírás***A DataItem komponensben importáld a PropTypes-t, és állítsd be a prop típusait:
  + Exportálás
    - Létrehozott fájl exportálása alul:
      * **export default DataItem;**

1. **Adatok lekérése a létrehozott komponenssel**
   * Szükséges React hook-ok (useState, useEffect) i mportálása (App.jsx):
     + import React, { useState, useEffect } from 'react';
   * Állapotváltozó definiálása az adatok tárolására:
     + const [items, setItems] = useState([]);
   * API URL definiálása:
     + const url = "http://localhost:8000/api/your\_endpoint";
   * Aszinkron függvény létrehozása az adatok betöltésére:
     + const fetchItems = async () => {
     + const response = await fetch(url);
     + const data = await response.json();
     + setItems(data.data);
     + // Ha a válasz { data: [...] } formátumú
     + };
   * **useEffect** hook használata az adatok betöltésére az első renderelés után:
     + useEffect(() => { fetchItems();}, []);

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

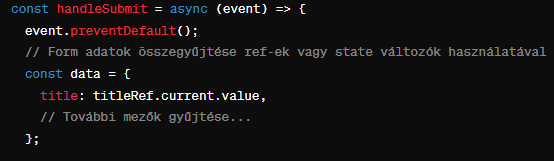
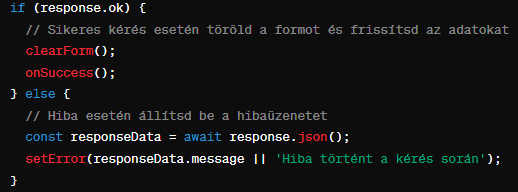
* + **DataItem** komponensek renderelése a **map()** függvény segítségével:

1. **Űrlap kialakítása**
   * DataForm komponens létrehozása (DataForm.jsx):
     + Hozz létre egy új funkcionális komponenst DataForm néven, amely az űrlapok kezelésére szolgál.
     + Importáld a **React**-et és a **PropTypes**-ot, valamint a **useRef** hook-ot:
       - **import React, { useRef } from 'react';**
       - **import PropTypes from 'prop-types';**
     + Az űrlap tartalmazzon input mezőket az általános adatok számára.
     + Referenciák használata az input mezőkben:
       - Referenciák definiálása az egyes input mezők számára:
       - const titleRef = useRef(null);
     + PropTypes beállítása:
       - A DataForm.propTypes segítségével az onSuccess prop típus megadása:
       - DataForm.propTypes = {

onSuccess: PropTypes.func.isRequired,

};

* + Input mezők és beküldés gomb:
    - Adj meg input mezőket a felhasználó által kitöltendő adatoknak, és egy beküldő gombot, amely elindítja a form beküldési folyamatát.
      * <input
      * type="text"
      * id="title"
      * className="form-control"
      * ref={titleRef}
      * />
  + Beküldés kezelése:
    - Definiálj egy handleSubmit függvényt, amely megakadályozza az űrlap alapértelmezett beküldését a event.preventDefault() hívásával.
    - A handleSubmit hívja meg az onSuccess callback-et, amely frissítheti az adatokat (például újra lekérheti azokat az API-tól).
      * const handleSubmit = (event) => {
      * event.preventDefault(); onSuccess(); };
  + Komponens használata az App komponensben:
    - Az App komponensben importáld és használd a DataForm komponenst, átadva neki az onSuccess propként a fetchItems függvényt.
    - Amikor a felhasználó beküldi az űrlapot és a kérés sikeres, a fetchItems függvény automatikusan lefut, frissítve az adatokat.
  + Plusz lehetőségek:
    - Propok fogadása destrukturálással:
      * function Name(props) {
      * const { propNeve, masikProp, megEgyProp } = props;
      * // Komponens logikája...
      * }
    - Ebben az esetben a PropTypes-t minden prop ellenőrzésére használni kell!

1. **Űrlap logikája (Submit)**
   * useState hook készítése az error state tárolására:
     + const [error, setError] = useState('');
   * Aszinkron kérés funkció létrehozása:
     + egy aszinkron funkciót definiálása a form adatok elküldésére:
     + A képen szöveg, képernyőkép, Betűtípus látható

       Automatikusan generált leírásÉrdemes **try-catch** blokkot alkalmazni

* Form tisztítás funkció:
  + funkció készítése, amely nullázza a form input mezőit sikeres submit után:
    - const clearForm = () => {
    - titleRef.current.value = '';
    - // További mezők törlése...
    - };
* Error megjelenítése az űrlapon:
  + A képen szöveg, képernyőkép, Betűtípus látható

    Automatikusan generált leírásAz űrlap visszatérési értékében a feltételes renderelés használata az error üzenet megjelenítésére, ha az nem üres: